2N－4

ノイマン型コンピュータの データ駆動型マシン技術を用いた高速化

～モジュールフェッチを前提とする ループ・アンローリングのハードウエアによる実現～

小椋 祐治，高橋 隆一，吉田 典可

広島市立大学 情報科学部 情報工学科

1 はじめに

IBM360/91の浮動小数点演算ユニットにおけるTomasuloのアルゴリズムは事実上のデータ駆動型マシン技術であったというのが筆者らの認識である[9].

本稿では，モジュールフェッチを前提にして，これまでコンパイラで行なわれてきたループアンローリングと等価なことを動的に行なう技術を考察する.

2 モジュールフェッチ

2.1 データ結合と機能的強度

プログラム設計の工程におけるモジュール設計では，モジュールの結合度はできるかぎり弱く，モジュールの強度はできるかぎり強いことが望ましいとされている[6]. 最も弱い結合度であるデータ結合は構造を持たない引数(単純な変数や配列など)を使っている場合である. 最も強い強度である機能的強度は独立したひとつの固有の機能だけでモジュールが構成された場合の強度である.

2.2 メモリバンド幅

ノイマン型コンピュータにおいては，十分なメモリバンド幅を確保できなければ性能のボトルネックとなる. 本稿では上述したようなモジュール設計が行なわれていることを前提に，メモリバンド幅はプログラムモジュール単位でフェッチすることができる程度であるとする.

3 ループアンローリング

以下の3つの場合に分けて，アンローリングを考察する.

3.1 ループ依存がない場合

以下は関数 $f(x) = x^2 + x + 1\ (0 \leq x \leq 100)$ の計算である:

```
LOOP : ADDI T2, T1, #1      // x + 1
       MULT T3, T1, T1      // x * x
       ADD  T4, T2, T3      // f(x)
       ADDI T1, T1, #1
       SUBI T5, T1, #100
       BNEZ T5, LOOP
```

コンパイラによるループアンローリングは，データ依存と制御依存によるパイプラインストールの解消が目的である[4]:

```
LOOP : ADDI T5, T1, #1      // x1 = x + 1
       ADDI T9, T1, #2      // x2 = x + 2
       ADDI T2, T1, #1      // x0 + 1
       ADDI T6, T5, #1      // x1 + 1
       ADDI T10, T9, #1     // x2 + 1
       MULT T3, T1, T1      // x0 * x0
       MULT T7, T6, T6      // x1 * x1
       MULT T11, T10, T10   // x2 * x2
       ADD  T4, T2, T3      // f(0)
       ADD  T8, T6, T6      // f(1)
       ADD  T12, T10, T11   // f(2)
       ADDI T1, T1, #3
       SUBI T13, T1, #100
       BNEZ T13, LOOP
```

データ駆動型の演算回路を有する場合はイテレーションの全てを並列に実行できる. 図1にハードウエアで展開されたデータフローグラフを示す.

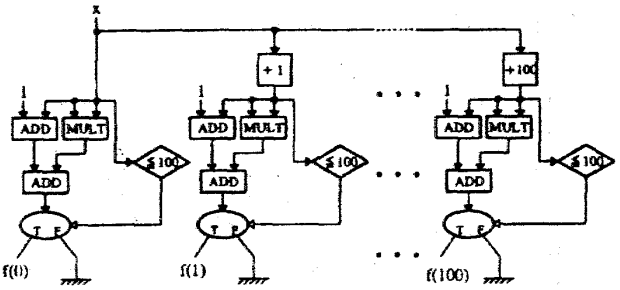

図1：データフローグラフの展開例1

3.2 ループ依存があり，かつイテレーション内にループとは独立に計算できる部分がある場合

関数 $f(x+1) = g(x) * f(x)$ において，$g(x) = x^2$ であるとする:

```
for(x=0;x<100;x++)
  f[x+1] = x * x * f[x];
```

コンパイラは下記のような書き換えを行なう[7]:

```
for(x=0;x<100;x++)
  g[x]= x * x;
for(x=0;x<100;x++)
  f[x+1] = g[x] * f[x];
```

図2にデータ駆動型演算回路を前提とするデータフローグラフの展開を示す:

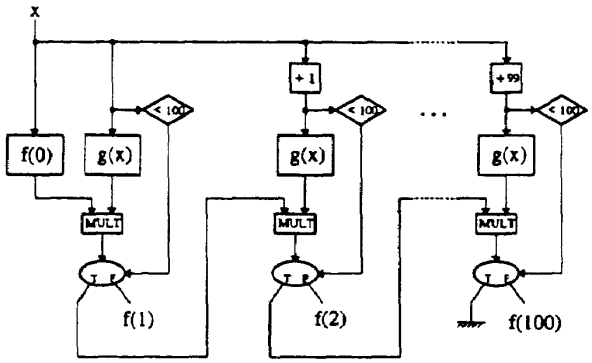

図2：データフローグラフの展開例2

全てのイテレーションにおいて $g(x)$ の計算は並列に行なわれる．そのため，$f(x-1)$ の値が生成されると即座に $f(x)$ の値が計算される．

3.3 ループ依存があり，イテレーション内にはループと独立に計算できる部分がない場合

以下はフィボナッチ数列の計算である:

```
for(i=0;i<=n-2;i++)
   x[i+2] = x[i+1] + x[i];
```

図3に展開された例を示す:

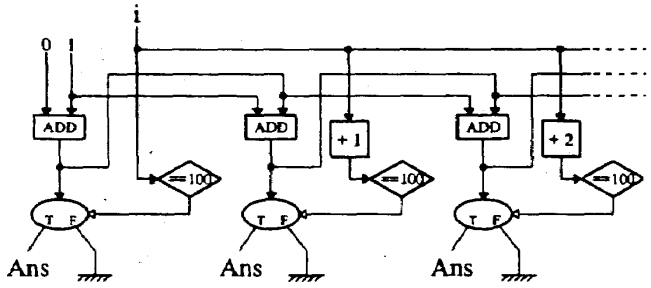

図3：データフローグラフの展開例3

コンパイラ技術を用いたパイプラインの実行は本質的には逐次実行である．データ駆動型マシンに対するループの展開は，データ依存関係の限界で処理を行なうことができる演算回路が前提である．命令レベルの並列性が増加すればそれだけプログラムの実行は高速になる．

4 マシンの構成

命令間で受渡しされるデータパケットのフォーマットを図4に示す:

module identifier	loop identifier	iteration identifier	tag field	data field

図4：パケットフォーマット

ユニット構成図を図5に示す:

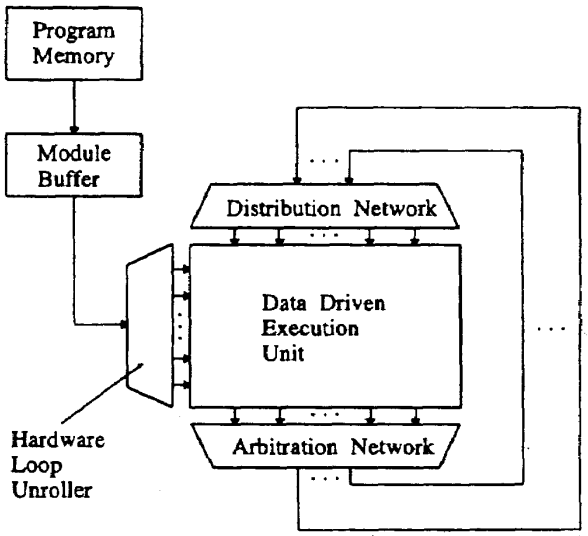

図5：ユニット構成図

データ依存関係はプログラムモジュールの間でも同様に議論できる[6]．モジュールのフェッチ，デコード，実行はパイプライン処理を想定している．

5 まとめ

本稿では，ノイマン型コンピュータの演算回路にデータ駆動型マシン技術を用いて高速化を図る手法を提案した．実現方法の詳細が今後の課題である．

参考文献

[1] J.A. シャープ 著，富田眞治 訳： データ・フロー・コンピューティング，サイエンス社 (1982)
[2] 雨宮真人，田中譲： コンピュータアーキテクチャ，オーム社，pp.133-172 (1988)
[3] Philip C.Treleaven et al.: Data-Driven and Demand-Driven Computer Architecture, Computing Surveys, Vol.14, Mar.1982, pp.93-143 (1982)
[4] David A.Patterson, John L.Hennessy : Computer Architecture A Quantitative Approach 2nd edition, Morgan Kaufmann Publishes, Inc. (1996)
[5] Kai Hwang, Fayé A.Briggs : Computer Architecture and Parellel Processing, McGraw-Hill, Inc. (1984)
[6] 高橋隆一： システム開発と設計，サイエンス社 (1996)
[7] 島崎眞昭： スーパーコンピュータとプログラミング，共立出版，pp.32-88 (1989)
[8] Kim P.Gostelow et al. : The U-interpreter, IEEE COMPUTER, Vol.15, No.2, Feb.1982, pp.42-49 (1982)
[9] 高橋隆一，児島彰，上土井陽子，吉田典可： マイクロコンピュータ設計教育環境 City-1，情処研報，設計自動化研究報告 No.83-6, pp.41-48 (1997)